

Addition of decimation filter information to GCF data blocks

In order to facilitate unambiguous determination of the decimation filters used to get to a certain sample rate, additional information has been added into the header of GCF data. This is placed in the byte previously marked as 'reserved', from DM24 v.091, build 40 onwards.

The byte value is coded as a lookup reference in a table of all *theoretical* valid combinations. Note that at this time, only a subset of the entries in this table are actually possible.

This information does not apply to status blocks or auxilliary multiplexed data channels. At this time, the 'reserved' byte is still not used, and will have the value zero.

A value of zero indicates that the information is not available, or the blocks are generated by an older version of firmware that does not support this enhancement.

Values 240 – 255 are not used at this time, and are reserved for future use.

The following pages contain:

- Pascal sample code to generate lookup table for sample rates
- Pascal sample code to translate an index code to sample rates without lookup table
- Pascal sample code to translate an index code to decimation filters without lookup table
- Lookup table expressed as sample rates
- Lookup table expressed as decimation filters

Pascal sample code to generate lookup table for samplerates.

```
Type
  T4Bytes = record
    case integer of
      1 : (dw : dword);
      2 : (w : array[0..1] of word);
      4 : (b : array[0..3] of byte);
    end;
  TSampleRates = array[0..255] of T4Bytes;

var
  SpsTable : TSampleRates;

const
  Filters : array[0..6] of byte = (20,16,10,8,5,4,2);
var
  a,b,c,d : integer;
  s1,s2,s3 : integer;
  count : integer;

begin
  count := 0;
  for a := 0 to 6 do
  begin
    s1 := 2000 div filters[a];
    for b := 0 to 6 do
      if (s1 mod filters[b])=0 then
      begin
        s2 := s1 div filters[b];
        for c := 0 to 6 do
          if (s2 mod filters[c])=0 then
          begin
            s3 := s2 div filters[c];
            for d := 0 to 6 do
              if (s3 mod filters[d])=0 then
              begin
                inc(count);
                SpsTable[count].b[0] := s1;
                SpsTable[count].b[1] := s2;
                SpsTable[count].b[2] := s3;
                SpsTable[count].b[3] := s3 div filters[d];
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;
```

Pascal sample code to translate an index code to sample rates without lookup table.

```
procedure IndexToSPS(index : byte; var s1,s2,s3,s4 : integer);
const
  Filters : array[0..6] of byte = (20,16,10,8,5,4,2);
var
  a,b,c,d : integer;
  count : integer;
begin
  count := 0;
  for a := 0 to 6 do
  begin
    s1 := 2000 div filters[a];
    for b := 0 to 6 do
      if (s1 mod filters[b])=0 then
      begin
        s2 := s1 div filters[b];
        for c := 0 to 6 do
          if (s2 mod filters[c])=0 then
          begin
            s3 := s2 div filters[c];
            for d := 0 to 6 do
              if (s3 mod filters[d])=0 then
              begin
                s4 := s3 div filters[d];
                inc(count);
                if count=index then exit;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;
```

Pascal sample code to translate an index code to decimation filters without lookup table.

```
procedure IndexToFilter(index : byte; var f1,f2,f3,f4 : integer);
const
  Filters : array[0..6] of byte = (20,16,10,8,5,4,2);
var
  a,b,c,d : integer;
  s1,s2,s3,s4 : integer;
  count : integer;
begin
  count := 0;
  for a := 0 to 6 do
  begin
    s1 := 2000 div filters[a];
    for b := 0 to 6 do
      if (s1 mod filters[b])=0 then
      begin
        s2 := s1 div filters[b];
        for c := 0 to 6 do
          if (s2 mod filters[c])=0 then
          begin
            s3 := s2 div filters[c];
            for d := 0 to 6 do
              if (s3 mod filters[d])=0 then
              begin
                inc(count);
                if count=index then
                begin
                  f1 := filters[a];
                  f2 := filters[b];
                  f3 := filters[c];
                  f4 := filters[d];
                  exit;
                end;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;
```

Lookup table expressed as sample rates.
 Highlighted entries indicate current capabilities.

0: -- not available --	60: 250 25 5 1	120: 400 100 10 1	180: 1000 50 5 1
1: 100 10 2 1	61: 250 50 5 1	121: 400 100 10 2	181: 1000 50 10 1
2: 100 10 5 1	62: 250 50 10 1	122: 400 100 10 5	182: 1000 50 10 2
3: 100 20 2 1	63: 250 50 10 2	123: 400 100 20 1	183: 1000 50 10 5
4: 100 20 4 1	64: 250 50 10 5	124: 400 100 20 2	184: 1000 50 25 5
5: 100 20 4 2	65: 250 50 25 5	125: 400 100 20 4	185: 1000 100 5 1
6: 100 20 5 1	66: 250 125 25 5	126: 400 100 20 5	186: 1000 100 10 1
7: 100 20 10 1	67: 400 20 2 1	127: 400 100 20 10	187: 1000 100 10 2
8: 100 20 10 2	68: 400 20 4 1	128: 400 100 25 5	188: 1000 100 10 5
9: 100 20 10 5	69: 400 20 4 2	129: 400 100 50 5	189: 1000 100 20 1
10: 100 25 5 1	70: 400 20 5 1	130: 400 100 50 10	190: 1000 100 20 2
11: 100 50 5 1	71: 400 20 10 1	131: 400 100 50 25	191: 1000 100 20 4
12: 100 50 10 1	72: 400 20 10 2	132: 400 200 10 1	192: 1000 100 20 5
13: 100 50 10 2	73: 400 20 10 5	133: 400 200 10 2	193: 1000 100 20 10
14: 100 50 10 5	74: 400 25 5 1	134: 400 200 10 5	194: 1000 100 25 5
15: 100 50 25 5	75: 400 40 2 1	135: 400 200 20 1	195: 1000 100 50 5
16: 125 25 5 1	76: 400 40 4 1	136: 400 200 20 2	196: 1000 100 50 10
17: 200 10 2 1	77: 400 40 4 2	137: 400 200 20 4	197: 1000 100 50 25
18: 200 10 5 1	78: 400 40 5 1	138: 400 200 20 5	198: 1000 125 25 5
19: 200 20 2 1	79: 400 40 8 1	139: 400 200 20 10	199: 1000 200 10 1
20: 200 20 4 1	80: 400 40 8 2	140: 400 200 25 5	200: 1000 200 10 2
21: 200 20 4 2	81: 400 40 8 4	141: 400 200 40 2	201: 1000 200 10 5
22: 200 20 5 1	82: 400 40 10 1	142: 400 200 40 4	202: 1000 200 20 1
23: 200 20 10 1	83: 400 40 10 2	143: 400 200 40 5	203: 1000 200 20 2
24: 200 20 10 2	84: 400 40 10 5	144: 400 200 40 8	204: 1000 200 20 4
25: 200 20 10 5	85: 400 40 20 1	145: 400 200 40 10	205: 1000 200 20 5
26: 200 25 5 1	86: 400 40 20 2	146: 400 200 40 20	206: 1000 200 20 10
27: 200 40 2 1	87: 400 40 20 4	147: 400 200 50 5	207: 1000 200 25 5
28: 200 40 4 1	88: 400 40 20 5	148: 400 200 50 10	208: 1000 200 40 2
29: 200 40 4 2	89: 400 40 20 10	149: 400 200 50 25	209: 1000 200 40 4
30: 200 40 5 1	90: 400 50 5 1	150: 400 200 100 5	210: 1000 200 40 5
31: 200 40 8 1	91: 400 50 10 1	151: 400 200 100 10	211: 1000 200 40 8
32: 200 40 8 2	92: 400 50 10 2	152: 400 200 100 20	212: 1000 200 40 10
33: 200 40 8 4	93: 400 50 10 5	153: 400 200 100 25	213: 1000 200 40 20
34: 200 40 10 1	94: 400 50 25 5	154: 400 200 100 50	214: 1000 200 50 5
35: 200 40 10 2	95: 400 80 4 1	155: 500 25 5 1	215: 1000 200 50 10
36: 200 40 10 5	96: 400 80 4 2	156: 500 50 5 1	216: 1000 200 50 25
37: 200 40 20 1	97: 400 80 5 1	157: 500 50 10 1	217: 1000 200 100 5
38: 200 40 20 2	98: 400 80 8 1	158: 500 50 10 2	218: 1000 200 100 10
39: 200 40 20 4	99: 400 80 8 2	159: 500 50 10 5	219: 1000 200 100 20
40: 200 40 20 5	100: 400 80 8 4	160: 500 50 25 5	220: 1000 200 100 25
41: 200 40 20 10	101: 400 80 10 1	161: 500 100 5 1	221: 1000 200 100 50
42: 200 50 5 1	102: 400 80 10 2	162: 500 100 10 1	222: 1000 250 25 5
43: 200 50 10 1	103: 400 80 10 5	163: 500 100 10 2	223: 1000 250 50 5
44: 200 50 10 2	104: 400 80 16 1	164: 500 100 10 5	224: 1000 250 50 10
45: 200 50 10 5	105: 400 80 16 2	165: 500 100 20 1	225: 1000 250 50 25
46: 200 50 25 5	106: 400 80 16 4	166: 500 100 20 2	226: 1000 250 125 25
47: 200 100 5 1	107: 400 80 16 8	167: 500 100 20 4	227: 1000 500 25 5
48: 200 100 10 1	108: 400 80 20 1	168: 500 100 20 5	228: 1000 500 50 5
49: 200 100 10 2	109: 400 80 20 2	169: 500 100 20 10	229: 1000 500 50 10
50: 200 100 10 5	110: 400 80 20 4	170: 500 100 25 5	230: 1000 500 50 2
51: 200 100 20 1	111: 400 80 20 5	171: 500 100 50 5	231: 1000 500 100 5
52: 200 100 20 2	112: 400 80 20 10	172: 500 100 50 10	232: 1000 500 100 10
53: 200 100 20 4	113: 400 80 40 2	173: 500 100 50 25	233: 1000 500 100 20
54: 200 100 20 5	114: 400 80 40 4	174: 500 125 25 5	234: 1000 500 100 25
55: 200 100 20 10	115: 400 80 40 5	175: 500 250 25 5	235: 1000 500 100 50
56: 200 100 25 5	116: 400 80 40 8	176: 500 250 50 5	236: 1000 500 125 25
57: 200 100 50 5	117: 400 80 40 10	177: 500 250 50 10	237: 1000 500 250 25
58: 200 100 50 10	118: 400 80 40 20	178: 500 250 50 25	238: 1000 500 250 50
59: 200 100 50 25	119: 400 100 5 1	179: 500 250 125 25	239: 1000 500 250 125

Lookup table expressed as decimation filters.
 Highlighted entries indicate current capabilities.

0: -- not available --	60: 8 10 5 5	120: 5 4 10 10	180: 2 20 10 5
1: 20 10 5 2	61: 8 5 10 5	121: 5 4 10 5	181: 2 20 5 10
2: 20 10 2 5	62: 8 5 5 10	122: 5 4 10 2	182: 2 20 5 5
3: 20 5 10 2	63: 8 5 5 5	123: 5 4 5 20	183: 2 20 5 2
4: 20 5 5 4	64: 8 5 5 2	124: 5 4 5 10	184: 2 20 2 5
5: 20 5 5 2	65: 8 5 2 5	125: 5 4 5 5	185: 2 10 20 5
6: 20 5 4 5	66: 8 2 5 5	126: 5 4 5 4	186: 2 10 10 10
7: 20 5 2 10	67: 5 20 10 2	127: 5 4 5 2	187: 2 10 10 5
8: 20 5 2 5	68: 5 20 5 4	128: 5 4 4 5	188: 2 10 10 2
9: 20 5 2 2	69: 5 20 5 2	129: 5 4 2 10	189: 2 10 5 20
10: 20 4 5 5	70: 5 20 4 5	130: 5 4 2 5	190: 2 10 5 10
11: 20 2 10 5	71: 5 20 2 10	131: 5 4 2 2	191: 2 10 5 5
12: 20 2 5 10	72: 5 20 2 5	132: 5 2 20 10	192: 2 10 5 4
13: 20 2 5 5	73: 5 20 2 2	133: 5 2 20 5	193: 2 10 5 2
14: 20 2 5 2	74: 5 16 5 5	134: 5 2 20 2	194: 2 10 4 5
15: 20 2 2 5	75: 5 10 20 2	135: 5 2 10 20	195: 2 10 2 10
16: 16 5 5 5	76: 5 10 10 4	136: 5 2 10 10	196: 2 10 2 5
17: 10 20 5 2	77: 5 10 10 2	137: 5 2 10 5	197: 2 10 2 2
18: 10 20 2 5	78: 5 10 8 5	138: 5 2 10 4	198: 2 8 5 5
19: 10 10 10 2	79: 5 10 5 8	139: 5 2 10 2	199: 2 5 20 10
20: 10 10 5 4	80: 5 10 5 4	140: 5 2 8 5	200: 2 5 20 5
21: 10 10 5 2	81: 5 10 5 2	141: 5 2 5 20	201: 2 5 20 2
22: 10 10 4 5	82: 5 10 4 10	142: 5 2 5 10	202: 2 5 10 20
23: 10 10 2 10	83: 5 10 4 5	143: 5 2 5 8	203: 2 5 10 10
24: 10 10 2 5	84: 5 10 4 2	144: 5 2 5 5	204: 2 5 10 5
25: 10 10 2 2	85: 5 10 2 20	145: 5 2 5 4	205: 2 5 10 4
26: 10 8 5 5	86: 5 10 2 10	146: 5 2 5 2	206: 2 5 10 2
27: 10 5 20 2	87: 5 10 2 5	147: 5 2 4 10	207: 2 5 8 5
28: 10 5 10 4	88: 5 10 2 4	148: 5 2 4 5	208: 2 5 5 20
29: 10 5 10 2	89: 5 10 2 2	149: 5 2 4 2	209: 2 5 5 10
30: 10 5 8 5	90: 5 8 10 5	150: 5 2 2 20	210: 2 5 5 8
31: 10 5 5 8	91: 5 8 5 10	151: 5 2 2 10	211: 2 5 5 5
32: 10 5 5 4	92: 5 8 5 5	152: 5 2 2 5	212: 2 5 5 4
33: 10 5 5 2	93: 5 8 5 2	153: 5 2 2 4	213: 2 5 5 2
34: 10 5 4 10	94: 5 8 2 5	154: 5 2 2 2	214: 2 5 4 10
35: 10 5 4 5	95: 5 5 20 4	155: 4 20 5 5	215: 2 5 4 5
36: 10 5 4 2	96: 5 5 20 2	156: 4 10 10 5	216: 2 5 4 2
37: 10 5 2 20	97: 5 5 16 5	157: 4 10 5 10	217: 2 5 2 20
38: 10 5 2 10	98: 5 5 10 8	158: 4 10 5 5	218: 2 5 2 10
39: 10 5 2 5	99: 5 5 10 4	159: 4 10 5 2	219: 2 5 2 5
40: 10 5 2 4	100: 5 5 10 2	160: 4 10 2 5	220: 2 5 2 4
41: 10 5 2 2	101: 5 5 8 10	161: 4 5 20 5	221: 2 5 2 2
42: 10 4 10 5	102: 5 5 8 5	162: 4 5 10 10	222: 2 4 10 5
43: 10 4 5 10	103: 5 5 8 2	163: 4 5 10 5	223: 2 4 5 10
44: 10 4 5 5	104: 5 5 5 16	164: 4 5 10 2	224: 2 4 5 5
45: 10 4 5 2	105: 5 5 5 8	165: 4 5 5 20	225: 2 4 5 2
46: 10 4 2 5	106: 5 5 5 4	166: 4 5 5 10	226: 2 4 2 5
47: 10 2 20 5	107: 5 5 5 2	167: 4 5 5 5	227: 2 2 20 5
48: 10 2 10 10	108: 5 5 4 20	168: 4 5 5 4	228: 2 2 10 10
49: 10 2 10 5	109: 5 5 4 10	169: 4 5 5 2	229: 2 2 10 5
50: 10 2 10 2	110: 5 5 4 5	170: 4 5 4 5	230: 2 2 10 2
51: 10 2 5 20	111: 5 5 4 4	171: 4 5 2 10	231: 2 2 5 20
52: 10 2 5 10	112: 5 5 4 2	172: 4 5 2 5	232: 2 2 5 10
53: 10 2 5 5	113: 5 5 2 20	173: 4 5 2 2	233: 2 2 5 5
54: 10 2 5 4	114: 5 5 2 10	174: 4 4 5 5	234: 2 2 5 4
55: 10 2 5 2	115: 5 5 2 8	175: 4 2 10 5	235: 2 2 5 2
56: 10 2 4 5	116: 5 5 2 5	176: 4 2 5 10	236: 2 2 4 5
57: 10 2 2 10	117: 5 5 2 4	177: 4 2 5 5	237: 2 2 2 10
58: 10 2 2 5	118: 5 5 2 2	178: 4 2 5 2	238: 2 2 2 5
59: 10 2 2 2	119: 5 4 20 5	179: 4 2 2 5	239: 2 2 2 2